Received September 24, 2021, accepted December 22, 2021, date of publication December 31, 2021, date of current version January 20, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3140015

Generative Deep Learning to Detect Cyberattacks for the IoT-23 Dataset

N. ABDALGAWAD, (Student Member, IEEE), A. SAJUN, Y. KADDOURA, (Student Member, IEEE), I. A. ZUALKERNAN, (Member, IEEE), AND F. ALOUL, (Senior Member, IEEE)
Department of Computer Science and Engineering, American University of Sharjah, Sharjah, United Arab Emirates

Corresponding author: N. Abdalgawad (g00068826@alumni.aus.edu)

This work was supported by the Open Access Program through the American University of Sharjah.

ABSTRACT The rapid growth of Internet of Things (IoT) is expected to add billions of IoT devices connected to the Internet. These devices represent a vast attack surface for cyberattacks. For example, these IoT devices can be infected with botnets to enable Distributed Denial of Service (DDoS) attacks. Signature-based intrusion detection systems are traditional countermeasures for such attacks. However, these methods rely on human experts and are time-consuming in terms of updates and may not exhaust all attack types especially zero-day attacks. Deep learning has shown some promise in intrusion detection. This paper shows that it is possible to use generative deep learning methods like Adversarial Autoencoders (AAE) and Bidirectional Generative Adversarial Networks (BiGAN) to detect intruders based on an analysis of the network data. The recently posted full IoT-23 dataset based on Somfy door lock, Philips Hue and Amazon Echo devices was used to train generative deep learning models to detect a variety of attacks like DDoS, and various botnets like Mirai, Okiruk and Torii. Over 1.8 million network flows were used to train the various models. The resulting generative models outperform traditional machine learning techniques like Random Forests. Both AAE and BiGAN-based models were able to achieve an F1-Score of 0.99. A BiGAN to detect unknown attacks was also trained to detect novel zero-day attacks with an F1-Score from 0.85 to 1.

INDEX TERMS Adversarial autoencoders, cyber security, generative adversarial networks, Internet of Things, intrusion detection systems.

I. INTRODUCTION

Internet of things (IoT) is one of the leading technologies today and is considered a natural extension of the internet by incorporating machine to machine communications and sensors. IoT applications have appeared in a variety of domains including health care, fitness, home energy management, classroom automation, smart cities and many more [1]. A typical IoT application consists of three layers; the perception layer, network layer and application layer. The perception layer is responsible for sensing and gathering information on the environment and sending it to the network layer. For example, surveillance cameras are one type of sensor recognizing unusual events like movement using sensors. The network/transport layer is considered a link between the perception layer and the cloud. This layer consists of many internet protocols and has to integrate communication technologies for information exchange such as Zigbee, 5G, MQTT, and Wi-Fi [2]. For example, a surveillance camera may use the home router and Wi-Fi to send a motion detection event to the main server. The application later utilizes the data received from the network layer to provide any operations or services required by users [3]. For example, the cloud service may send an alarm to a mobile application being used a home owner indicating that a movement was detected in one of their surveillance cameras.

IoT is vulnerable to security risks at every architectural layer and has faced security challenges since its emergence [4]. For example, Butt *et al.* [5] examined the type of attacks in Smart Health Systems and found that the attacks are Denial of Service Attack (DoS), Fingerprint and Timing-based Snooping (FATS), Router Attack, Select Forwarding (SF) Attack, Sensor Attack and Replay Attack. In general, the perception layer can suffer from attacks such as malicious code injection, eavesdropping and interference [3], [6]. Similarly, the network layer is susceptible to attacks like

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Huei Cheng.

spoofing, denial of service, man-in-the-middle and routing information [6]. Privacy is another major concern [4]. IoT devices require strong authentication systems, which a lot of IoT devices do not have due to resource constraints like CPU or power/battery limitations [7]. Finally, the application layer is also open to attacks from viruses, worms and phishing attacks. Andrea *et al.* [8] classified such attacks into four categories; physical, software, network and encryption. The physical attack occurs when the perpetrator is close to the system physically, while the software attack is when the device contracts a bug that allows unauthorized access to the device that can harm the system. The network attack occurs when the IoT network is accessed to manipulate a device to cause damage and the encryption attack takes place when the IoT encryption is compromised.

Botnet is a specific attack mechanism exploiting IoT devices. Angrishi *et al.* [9] describes a botnet as a large group of internet-enabled devices that are controlled in order to make simultaneous requests to a specified server (or group of servers) to overwhelm it and prevent it from responding to legitimate requests, thereby essentially stopping its service. This attack is a distributed denial of service (DDoS) attack. Two methods used in such DDoS attack: reflection and amplification. Both lead to an exhaustion of the bandwidth and resources of the target. And due to the increase in sophistication of these attacks, they are very hard to identify [9]. IoT botnets are not only a threat to IoT device owners but also to anyone on the internet. Because DDoS attacks need a significant network traffic to compromise services, and IoT devices provide a perfect host due to the sheer number of IoT devices available and in use today as well as their generally poor security, making them the low-hanging fruit [10].

The earliest known botnet was Linux/Hydra that was released in 2008. It had both spreading features and ability to launch DDoS. The Psyb0t in 2009 targeted routers and modems, compromising almost 100 thousand devices. The infection methods used brute force attacks using 6000 predefined usernames and 13 thousand predefined passwords [9]. Linux.Darlloz is another example of an IoT botnet that infected more than 31 thousand devices and was an IoT worm. After infecting the IoT device, it would prevent any users from accessing the device by dropping telnet traffic and terminating the telnetd process [9]. Spike (Dofloo) is another botnet that was targeting windows and Linux based PCs and was using in launching several attacks on organizations in Asia and the US. It had a peak of 215Gbps, allowing it to launch several types of attacks for DDoS. BASHLITE is another botnet that controlled over a million IoT devices and could launch attacks at 400 Gbps [9].

Das *et al.* [11] describes Mirai, a recent Botnet in which a virus scans for vulnerable devices and attaches itself to them making them connected to Command-and-Control Servers (C&C servers). By being connected to C&C servers, they are vulnerable to attacks or can be used to attack other devices.

Das *et al.* [11] and Tushir *et al.* [12] stated that IoT devices connected to Mirai Botnets are mostly used to carry DDoS attacks against a target device. Tushir *et al.* [12] examined the effects of the Mirai attack in IoT devices and found that the energy consumption by IoT devices increases by around 40% and the storage used is increased by half. The extent of the danger that a botnet could pose was truly shown in 2016, when the Mirai botnet was unleashed. This botnet infected 4000 devices per hour and had around half a million active infected devices at a ground-breaking 1.1Tbps attack. The infected IoT devices were spread over 164 countries. Since then, there have been many Mirai botnet versions and variations such as Persirai, Hajime and BrickerBot [13]. DDoS attack victims included websites, cloud providers, individuals, colleges, telecommunication companies, DNS providers (Dyn) which offered services to multiple websites such as Reddit, Amazon, Spotify, Airbnb and others [9].

There is clearly a need to reinforce IoT security in order to prevent the development of such botnets, but also ensure that all possible DDoS victims are well-prepared in detecting such attacks, especially since advances in these botnets make them very hard to identify until it is too late. According to Statista [14], in 2021 there are almost 8.74 billion IoT connected devices worldwide and a Cisco white paper [15] estimates there will be around 30 billion connected devices in 2023 compared to around 18 in 2018. According to the same paper, it is expecting to have around 15 million DDoS attacks by 2023 compared to 7 million in 2018 [15].

One of the key methods of preventing such attacks is the deployment of a strong Intrusion Detection System (IDS) [16] that can detect any type of intrusion. Currently, IDS's use two primary methods for detecting attacks: signature-based and anomaly-based. Signature-based methods depend on the well-known attacks and their updates are time consuming. Anomaly-based attacks, on the other hand, are based on data and depend on the machine understanding the normal behavior and rejecting all incoming connections that seem abnormal. Developing an IDS for automatic detection of cyberattacks requires an appropriate dataset for training. Iot-23 Garcia *et al.* [17] is one such data set that has been recently released and specifically addresses cyberattacks involving IoT devices. This dataset was published early 2020.

This paper used the Iot-23 data set to explore the use of generative deep learning techniques that can automatically detect and classify IoT cyberattacks. The primary contribution of this paper is that it used the complete IoT-23 dataset for building an IDS and achieved state-of-the-art result in anomaly detection.

II. RELATED WORK

Much work has been done in building intrusion detection systems using a variety of machine learning and deep learning models. For example, Pang *et al.* [18] reviewed the different models that have been used for anomaly detection including Generative Adversarial Networks (GANS). Resende *et al.* [19] surveyed different random forest models used in intrusion detection systems with a variety of datasets using different features and classes.

Li *et al.* [20] proposed using convolutional neural networks (CNN) to classify the botnet attacks. They split the dataset into 4 separate parts (according to the correlations between the features) and trained and tested the same model on the data separately for binary classification (CNN1, CNN2, CNN3, CNN4). Another model called CNN0 was trained using the complete data. All models were trained on the NSL-KDD dataset [21] and tested on the KDDTest+ and KDDTest-21. The highest accuracy was obtained by the CNN1 model on both testing sets yielding 82.62% and 67.22% accuracy respectively. An ensemble of all models resulted in accuracies of 86.95% and 76.67% on both testing datasets respectively.

Latif *et al.* [22] presented a deep random neural network (DRaNN) and trained it using the UNSW-NB15 dataset [23]. Their model achieved an accuracy of 99.54%.

Xu *et al.* [24] proposed an autoencoder (AE) model based on Long Short-Term Memory (LSTM) for detecting intrusions. They trained the model on 5 datasets in total: ARP, Fuzzing, Mirai, SSDP Flood, and Video Injection from the Mirsky team [25]. They trained their model on each dataset separately with a binary classification. They then compared the results of their model with two traditional machine learning methods. The results of their model ranged between F1-scores of 92.4-96.8 with the Mirai performing the best at 99.6. Their model generally outperformed Support Vector Machine (SVM) and K-Nearest Neighbors (KNN), an Autoencoder (AE) and a stacked auto-coder.

Shahriar *et al.* [26] proposed a G-IDS framework that included 4 segments: database module, IDS module, controller module and synthesizer module. The database module collects real intrusion detection data as well as synthesized data from the GAN/synthesizer module, each with a flag to distinguish the data sources. The synthesized data can be either pending, which cannot be used until further notice, or synthetic which is verified generated data that stays in the database. The controller module inspects the pending data and checks if it contributes to the performance of the IDS and if it does, the flag is changed to synthetic. Their core was an ML-based IDS which was a multi-layer ANN model with 4 hidden layers 50 neurons each. It was trained twice, once with the pending data and once without to calculate the performance metrics. The GAN was then used mainly for the generator to produce real-like data to train the ANN model which was used to perform the multiclass classification. Most of the labels had relatively high F1 scores but a few had lower scores like 0.41 and 0.68.

Like most anomaly detection tasks, cyberattacks often result in unbalanced data. Fan *et al.* [27] built artificial anomalies based on known classes to test their model. Since the boundary between the normal data and the unknown anomaly is unknown and may be very close, they only changed the value of one feature randomly and kept the rest the same. They used the NSL-KDD'99 dataset [21] and their model is an inductive decision tree learner, RIPPER. They kept injecting new data into their training set and testing with new anomalies and their results showed an increase in the true detections from 59% up to 100%.

R.M. *et al.* [28] proposed a deep learning model to detect attacks in Internet of Medical Things (IoMT). The models involved a hybridization of Principal Component Analysis (PCA) technique and Grey wolf optimization metaheuristic algorithm. The model had an accuracy of 15% higher than existing models and a reduction in training time by 32%.

GANs have also been used in data generation and augmentation related to intrusion detection tasks. For example, Shahid *et al.* [29] used a GAN to build a sequence of packets. The dataset was built using Google Home Mini creating 42 packets in a sequence with vocabulary size 535. The model consisted of an autoencoder and a GAN. The encoder built the latent space from the sample and sent it to the GAN. The GAN tries to learn this latent space and generate samples based on it. Then the decoder takes in the latent space generated and convert it to actual text-based packets. Similarly, [30]–[32] used GANs to balance the datasets then train it using other models. This was mostly done by generating more samples of the minority class in order to balance it with the majority. For example, Salem *et al.* [30] used cycle-GAN and a Multi-Layer Perceptron for classification. The F1-score of traditional balancing methods like Synthetic Minority Over-Sampling Technique (SMOTE) performed poorly when compared with their model and cycle-GAN performed the best out of the three with an F1-Score of 41.64%.

GANs have also been used to directly implement intrusion detection systems. For example, Huang *et al.* [31] developed a system called Imbalanced Generative Adversarial Intrusion Detection System (IGAN-IDS). The proposed architecture had three modules: feature extraction, IGAN and the DNN modules. The feature extraction was a filter and consisted of an embedding with dimension of 356 and MLP which consisted of 2 fully connected layers both of dimensions 128 and output of sigmoid function. The IGAN modules was responsible for generating samples and had, for both discriminator and generator, a learning rate of 0.00005 and batch size of 128. The discriminator was a 3 fully connected MLP with dimensions 256, 128 and 64. The generator consisted of 3 fully connected MLP all with dimension 256, 64 kernels of size 16 that constituted the convolutional layers. The IGAN was fully optimized until the discriminator converged to 0.5. The DNN module was then trained on the newly generated samples. The DNN had 6 layers: fully connected layer of size 256 with sigmoid function, followed by 2 convolutional layers each of size 64 with ReLU function, followed by dropout layer with rate 0.2, followed by fully connected layer of size 32 with Leaky ReLU function, followed by a fully connected layer with the size of classes using a Softmax function. The experiment used 3 different intrusions datasets: NSL-KDD [21], UNSW-NB15 [23], CICIDS2017 with 5, 10 and 6 classes respectively [33]. The system achieved an accuracy of 84.45%, 82.53% and

99.79% respectively and F1-scores of 84.17, 82.86 and 99.79 respectively.

A similar work was reported by Yilmaz *et al.* [32] where they did binary classification on an imbalanced dataset having 5 hidden layers for both discriminator and generator with activation functions ReLU, Sigmoid and learning rates 0.0025 for discriminator and 0.02 for generator.

Chauhan *et al.* [34] used GANs to demonstrate that deep learning methods were not sufficient in the detection of new attack profiles. They first trained a GAN based on the CICIDS2017 [33] dataset and used the SHapley Additive exPlanations (SHAP) [35] method to extract features from the dataset based on their importance and impact on the output. The highest detection rate while training the model was 79%. After that they used two techniques to update the feature profile including increasing the number of features used and swapping the current features with others. This resulted in several adversarial attacks that went undetected with the detection rate decreasing to 5.23% and 3.89% respectively for each change.

Adversarial autoencoders (AAE) is an autoencoder based on a GAN; AAE's can reduce the probability of overfitting because it can influence the distribution approximated by the hidden layer as shown by Makhzani *et al.* [36] and Puuska *et al.* [37]. The encoder tries to generate samples based on the chosen distribution, while the decoder tries to recreate the original data from the latent space. The discriminator tries to know if the sample which was generated by the encoder, is in fact generated or from the chosen distribution. For example, Puuska *et al.* [37] used an adversarial autoencoder was applied to a DARPA intrusion dataset and the accuracy achieved was higher than that of normal autoencoder but based on anomaly detection.

Hara *et al.* [38] also used an AAE for intrusion detection. Their focus was to validate the implementation of semi-supervised learning through AAE and DNN by changing the percentage of labelled data. They use the NSL-KDD dataset [21], and concluded that the higher the labelled the data, the higher the accuracy. Their highest accuracy achieved was 83.11%.

The BiGAN architecture has also been used for intrusion detection. The difference between a regular GAN and a BiGAN is that the BiGAN has an encoder to map out the data back to latent space as shown in Kaplan *et al.* [39]. The generator converts some latent space z to fake data G(z), and the encoder converts the real data x into some represented latent space E(x). Unlike standard GANs, the discriminator also learns concatenated inputs mapped to the same dimension (z, G(z)) and (E(x), x). Donahue *et al.* [40] first evaluated the feature learning capabilities of the BiGAN by using unsupervised training then transferring the trained encoder to use in supervised learning tasks. They evaluated them using images in the ImageNet database. However, their maximum classification accuracy for the ImageNet was 56.2%. On the other hand, Donahue *et al.* [41] attempted to extend the state-of-the-art BigGAN to a BiGAN (BigBiGAN) on the same database. They trained their architecture using unsupervised learning and achieved an accuracy of 60.8% which improved previously published results from 55.4%.

Kaplan *et al.* [39] used a BiGAN for anomaly detection through training them on the normal data of the KDDCUP99 dataset [21] and placed samples of the attack classes in the test set. BiGAN with their proposed algorithm attained the best performance with an F1-score of 90.8. Alabugin *et al.* [42] used a similar approach using BiGAN on benign data they produced from their testbed.

TABLE 1. Summary of related work that used GANs.

Paper	Pros	Cons
Shahriar *et al.* [26]	The work used generative adversarial networks to generate samples for imbalanced datasets.	The work did not use GANs for classification.
Hara *et al.* [38]	The work used AAE to train their models.	Their highest achieved accuracy was 83.11%.
Kaplan *et al.* [39]	The work used BiGAN for anomaly detection using an intrusion detection related dataset.	The KDDCUP99 is proved to have redundancy issues [21].
Puuska *et al.* [37]	The work used AAE and their results showed that AAE had better performance than autoencoders.	The work did not consider classification and their highest accuracy was only 65%.
Chauhan *et al.* [34]	The work used GANs to generate new samples of data that could not be detected.	The work did not use GANs to classify samples.

Table 1 shows a summary of the most recent and related work. The table show pros and cons of each work. The cons do not necessarily mean an inherent issue with the work, but rather a research gap that is being addressed in this paper.

III. METHODOLOGY

A. THE IOT-23 DATASET

This paper used the IoT-23 dataset [17]. This data is based on the network traffic obtained from Internet of Things (IoT) devices with 20 malware and 3 benign captures. It is worth noting that the 3 benign captures were carried in three real IoT devices: Somfy door lock, Philips Hue and Amazon Echo. The 20 malware captures were captured using a Raspberry Pi.

In the IoT-23 data set, after the .pcap files were generated, they were passed through the Zeek network Analyzer to generate log files. One of the capabilities of Zeek is to produce connection log files that show the properties of a connection or a flow between two entities [43]. The .pcap files were analyzed manually to identify the properties of the different labels. Then, a python script was run through the log files to add labels based on the analysis. The malware files sizes varied from few kilo Bytes to about 10 Giga Bytes. The unit of analysis is therefore a flow.

Although the IoT-23 dataset is multi-labelled, the labels have similar classes. Labels are the different types of attacks, but classes can be a combination of different attacks. For example, a label could be C&C or PartOfAHorizontal-

PortScan and both have different meanings, whereas a class could be C&C-PartOfAHorizontalPortScan which means that both malware attacks are present for the flows in this class. The labels are described as below [17]:

- Attack: a type of attack from the infected device to another host where it tries to take an advantage of a vulnerability.
- Benign: no suspicious or malicious activities were found in the connections.
- C&C: the infected device was connected to a CC server.
- DDoS: a Distributed Denial of Service attack is being executed by the infected device.
- FileDownload: a file is being downloaded to our infected device.
- HeartBeat: the packets sent on this connection are used to keep a track on the infected host by the C&C server.
- Mirai: the connections have characteristics of a Mirai botnet.
- Okiru: the connections have characteristics of a Okiru botnet.
- PartOfAHorizontalPortScan: the connections are used to do a horizontal port scan to gather information to perform further attacks.
- Torii: the connections have characteristics of a Torii botnet.

The frequency of flows in classes obtained from the labelled data are shown in Table 2.

This dataset had 19 features .as shown in Table 3 ([44], [45]).

Certain features had values or letters with special meanings. These features were conn_state and history and their values' description can be seen in Table 4 and Table 5 respectively [45].

B. FEATURE SELECTION AND DATA CLEANING

Features 'local_orig', 'local_resp' which were empty for all the files and hence they were dropped. Based on previous work on pre-processing of intrusion detection datasets, features including IP addresses and port numbers were dropped. The 'history' feature was a sequence of values that describe the history of the connection was also initially dropped. Fig. 2 shows the correlation graph of the remaining features.

Based on the correlation graph, orig_pkts and orig_ip_bytes correlate, similarly with resp_pkts and resp_ip_bytes. The correlated features (orig_ip_bytes and resp_ip_bytes) we hence dropped. Fig. 1 shows boxplots of the remaining features and as can be seen in Fig. 1 (a), all the values were zero so this feature was dropped. The boxplots of the other features show variations between classes, so they are distinguishable. The final features included proto, service, duration, orig_bytes, resp_bytes, conn_state, orig_pkts and resp_pkts.

Extreme minority classes with less than 100 samples in the dataset were dropped; 'C&C-FileDownload', 'FileDownload', 'C&C-Torii', 'C&C-HeartBeat-FileDownload', 'PartOfAHorizontalPortScan-Attack', 'Okiru-Attack' and 'C&C-Mirai'. The features 'orig_bytes', 'resp_bytes' and 'duration' contained null values and based on the previous work, the null values were replaced with mean value of the respective features. Data type 'duration' feature was recoded from timedelta64 to time in seconds. Finally, after dropping several features, the remaining flows contained duplicates that were removed as well. After dropping the duplicate

TABLE 2. Number of flows for each class in the dataset.

Class	Number of flows
C&C-FileDownload	53
C&C	21995
Benign	30858735
DDoS	19538713
C&C-Torii	30
FileDownload	18
PartOfAHorizontalPortScan	213852924
Attack	9398
C&C-HeartBeat	33673
C&C-HeartBeat-FileDownload	11
C&C-HeartBeat-Attack	834
C&C-PartOfAHorizontalPortScan	888
Okiru	60990708
Okiru-Attack	3
C&C-Mirai	2
PartOfAHorizontalPortScan-Attack	5
Total	**325307990**

TABLE 3. Feature description of the IoT-23 dataset.

Feature		Description
1	uid	Unique ID
2	id.orig-h	Source IP address
3	id.orig-p	Source port
4	id.resp-h	Destination IP address
5	id.resp-p	Destination port
6	proto	Transaction protocol: icmp, udp, tcp
7	service	dhcp, dns, http, irc, ssh, ssl
8	duration	Total duration of flow
9	orig_bytes	Number of payload bytes the originator sent
10	resp_bytes	Number of payload bytes the responder sent
11	conn_state	Connection state. Possible values are found in Table IV
12	local_orig	T if the connection originated locally and F if it originated remotely
13	local_resp	T if the connection is responded locally and F if it is responded remotely
14	missed_bytes	Number of bytes missed in content gaps, which is representative of packet loss
15	history	State history of connections as a string of letters. The letter is uppercase if it comes from the responder and lowercase if it comes from the originator. Possible letters can be seen in Table V
16	orig_pkts	Number of packets that the originator sent
17	orig_ip_bytes	Number of IP level bytes that the originator sent
18	resp_pkts	Number of packets that the responder sent.
19	resp_ip_bytes	Number of IP level bytes that the responder sent

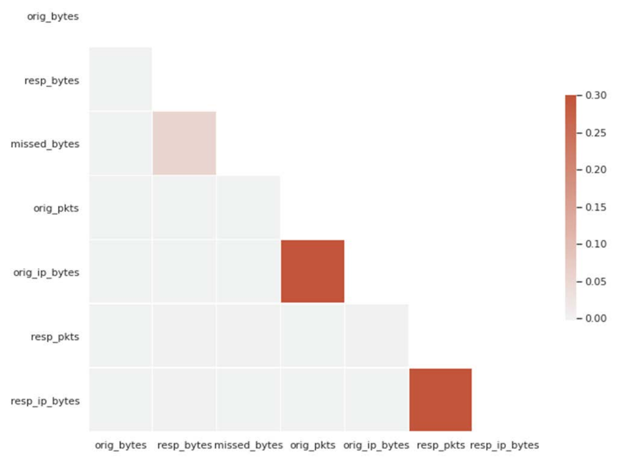

FIGURE 1. Correlation graph of the features.

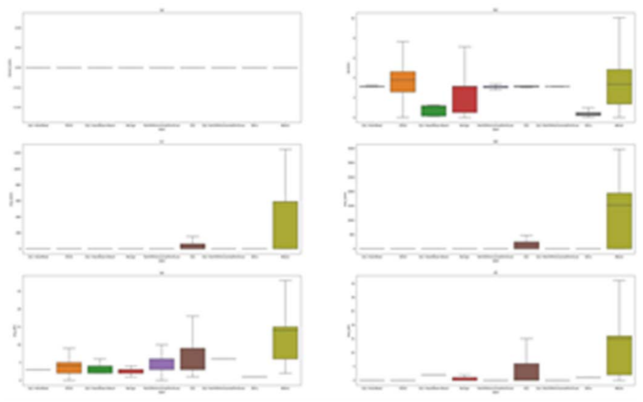

FIGURE 2. Boxplots of different features versus labels.

flows, Table 6 shows the final number of remaining flows in each class.

Preprocessing involves various tasks like feature selection, encoding, normalization, balancing, etc. Faker *et al.* [46] summarized some of the pre-processing techniques for network data including removing socket specific information such as IP addresses and port numbers, replacing missing values with mean value of the feature, replacing infinity values with the maximum value in the features, normalizing and encoding the data.

Categorical features like 'service', 'proto' and 'conn_state' required encoding. For example, Ieracitano *et al.* [47], Wu *et al.* [48], and Xiao *et al.* [49] used one-hot encoding to encode categorical data, whereas Zhang *et al.* [50] used dummy variable encoding. As per the previous work, the most common type of encoding was one-hot encoding and therefore, these three categorical features were one-hot-encoded. The data was then normalized between 0 and 1 using min-max scaling.

Since there was considerable data imbalance even after removing the extreme minority classes, common balancing techniques [51] were explored namely Random Over-Sampler, Random UnderSampler and Synthetic Minority Oversampling Technique (SMOTE) by Chawla *et al.* [52].

TABLE 4. Description of the conn_state values.

Value	Description
S0	Connection attempt seen, no reply
S1	Connection established, not terminated with no byte count
SF	Normal establishment and termination with byte count
REJ	Connection attempt rejected
S2	Connection established and close attempt by originator seen with no reply from responder
S3	Connection established and close attempt by responder seen with no reply from originator
RSTO	Connection established and originator aborted by sending a RST
RSTR	Responder sent a RST
RSTOS0	Originator sent a SYN followed by a RST and no SYN-ACK seen from the responder
RSTRH	Responder sent a SYN ACK followed by a RST, no SYN seen from the originator
SH	Originator sent a SYN followed by a FIN, no SYN ACK seen from the responder
SHR	Responder sent a SYN ACK followed by a FIN, no SYN seen from the originator
OTH	No SYN seen

TABLE 5. Description of history values.

Value	Description
s	a SYN w/o the ACK bit set
h	a SYN+ACK ("handshake")
a	a pure ACK
d	packet with payload ("data")
f	packet with FIN bit set
r	packet with RST bit set
c	packet with a bad checksum (applies to UDP too)
g	a content gap
t	packet with retransmitted payload
w	packet with a zero window advertisement
i	inconsistent packet (e.g. FIN+RST bits set)
q	multi-flag packet (SYN+FIN or SYN+RST bits set)
^	connection direction was flipped by Zeek's heuristic

TABLE 6. Number of each class after dropping duplicates.

Class	Number
Benign	113,860
DDoS	1,643,225
C&C-HeartBeat-Attack	743
C&C-PartOfAHorizontalPortScan	327
C&C-HeartBeat	10,239
PartOfAHorizontalPortScan	69,198
Okiru	14,942
Attack	9,363
C&C	8,939
Total	**1870836**

A combination of Random UnderSampling and SMOTE were used to under sample the majority class while oversampling the minority classes.

The balancing regime involved calculating 25% of the majority class and down sampling the majority class to this number. A threshold of 10% of this down sampled number is selected and the minority classes which fell below this threshold were up sampled to this number. This further helped reduce the imbalance ratio to 10:1.

C. INTRUSION DETECTION MODELS

Building intrusion detection systems using the IoT-23 data set has been done before. For example, using only the Mirai botnet samples from the IoT-23 dataset, Hussain *et al.* [53] investigated the possibility of a universal features set that would allow machine learning algorithms to classify Mirai botnet attacks regardless of the dataset used. They extracted features from the .pcap files using CICFlowmeter and found that top 6 features among several datasets. They created training and test sets of the ratio 80:20 and resulted in 100% accuracy using Random Forest. Similarly, Hegde *et al.* [54] focused on identifying botnets by running multiple machine learning and deep learning classifiers. They used the IoT-23 dataset (for botnets only) as well as some benign data they captured from a testbed environment. They created a small and large dataset with 95% benign data and 5% malicious. They achieved the highest accuracy of 99.9%. With four malware and three benign captures of the IoT-23 only, Dutta *et al.* [44] implemented a deep learning model for anomaly detection using a stacked generalization method leveraging a Deep Neural Network (DNN) and a Long Short-Term Memory (LSTM) with KFold cross validation. The highest F1-score achieved was 0.98 with an accuracy of 0.997. Finally, Kumar *et al.* [55] used the C&C samples of the IoT-23 dataset to verify an architecture.

This paper proposed and evaluated three generative deep learning models using the complete IoT-23 data set. Each model is described below.

D. ADVERSARIAL AUTOENCODER MODEL

Fig. 3 shows the proposed model. As the figure shows, an adversarial autoencoder [35] took an input x of feature size 27 and created a latent representation E(x) of size 6. Generator, G, generates the original data, x', from the latent features. The discriminator, D, takes in the generated features and a random z and distinguishes them. Intrusion detection is done by encoding the test data to the latent space and then training a classifier like KNN, for example, to identify the intrusion class (e.g., Mirai). This model is similar to the previous work in [56].

The original 27 features were used and reduced to a latent dimension of size 6 using the autoencoder. The AAE was trained for 1000 epochs with batch size of 10. Adam optimizer with a learning rate of 10^(−4). The loss equation used for the autoencoder was the mean square error as shown in equation (1). The loss function of the discriminator and the generator was the binary cross entropy which is shown in equation (2). Table 7, Table 8 and Table 9 show the characteristics of the layers of the encoder, decoder, and discriminator respectively.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(A_i - P_i)^2 \tag{1}$$

where $A_i = actual, P_i = predicted, n = data\ point$

$$\min_G \max_D E_{x\sim p_{\text{data}}}[logD(x)] + E_{z\sim p_{(z)}}[log(1 - D(G(z))] \tag{2}$$

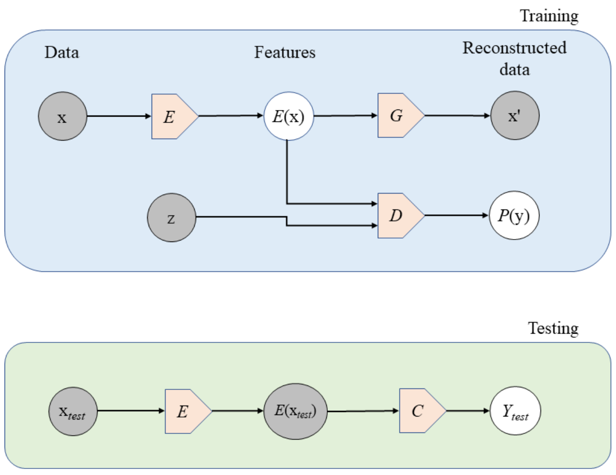

FIGURE 3. AEE and classifier (e.g. kNN).

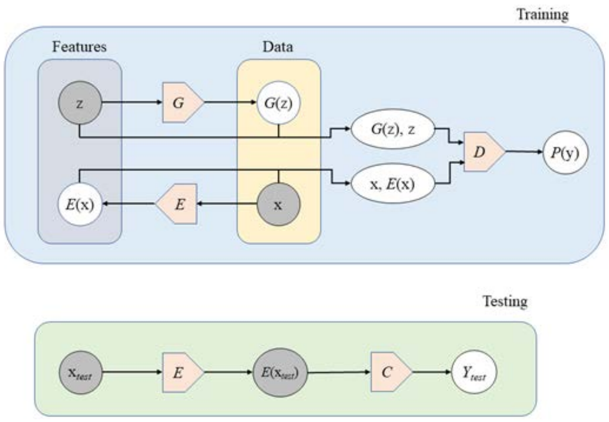

FIGURE 4. BiGAN and classifier (e.g. kNN).

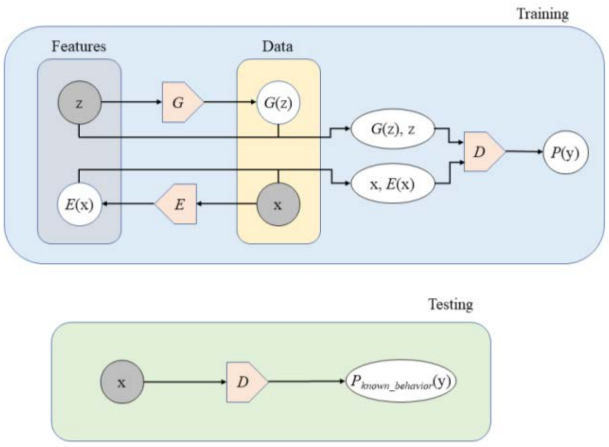

FIGURE 5. Known/Unknown BiGAN architecture.

E. BIDIRECTIONAL GENERATIVE ADVERSARIAL NETWORKS (BIGAN)

As Fig. 4 shows, a BiGAN was also used to create a latent representation of the input data. The encoder E took in all 27 features and produced their latent representation of size

TABLE 7. AAE encoder's layers characteristics.

Layer Number	Neurons	Input Dimension	Activation Function
1	128	41	Relu
2	128	-	Relu
3	16	-	-

TABLE 8. AAE decoder's layers characteristics.

Layer Number	Neurons	Input Dimension	Activation Function
1	128	16	Relu
2	128	-	Relu
3	41	-	Sigmoid

TABLE 9. AAE discriminator's layers characteristics.

Layer Number	Neurons	Input Dimension	Activation Function
1	128	16	Relu
2	128	-	Relu
3	1	-	Sigmoid

TABLE 10. BiGAN encoder's layers characteristics.

Layer Number	Type	Neurons	Input Dimension	Activation Function
1	Dense	512	27	LeakyRelu
2	BatchNormalization	-	-	-
3	Dense	512	-	LeakyRelu
4	BatchNormalization	-	-	-
5	Dense	8	-	-

8 and the generator G produced generated network data from the noise z of the same size as the latent representation. Both sets (input data, E (input data)) and (G (noise z), noise z)) were fed to the discriminator, D. The BiGAN was trained using 1,000 epochs with a batch size of 32. Adam optimizer with a learning rate of 0.0002 was used. The objective function of the BiGAN is shown in equation (3). Table 10, Table 11 and Table 12 describe the layers of the encoder, generator, and discriminator.

$$\min_{G,E} \max_{D} V(D, E, G) \tag{3}$$

where

$$V(D,E,G) := E_{x\sim p_x}\underbrace{\left[E_{z\sim p_E(\cdot|x)}[logD(x,z)]\right]}_{logD(x,E(x))} + E_{z\sim p_z}\underbrace{\left[E_{x\sim p_G(\cdot|z)}[log(1-D(x,z))]\right]}_{log(1-D(G(z),z))}$$

F. BIDIRECTIONAL GENERATIVE ADVERSARIAL NETWORKS (BIGAN) TO DETECT UNKNOWNS

A BiGAN that can detect unknown anomalies as well as its testing is shown in Fig. 5. The training architecture is similar to the previous GAN used. The testing architecture differs here where we are using the trained discriminator, D, to distinguish whether the input is known or unknown behaviour. The BiGAN was trained on all the data including benign and anomalies. In order to test if the BiGAN could detect unknown anomalies, synthetic unknown anomalies were created by randomly selecting a sub-set of features (1 or 2, for example) and then randomly changing the value of the features. The BiGAN were trained for 40,000 epochs with a batch size of 32 and used Adam optimizer with a learning rate of 0.003 and a latent space of 8. The architectures used for the encoder and the discriminator layers are as previously shown in Table 10 and Table 12. The layers of the generator are described in Table 13.

TABLE 11. BiGAN generators's layers characteristics.

Layer Number	Type	Neurons	Input Dimension	Activation Function
1	Dense	512	27	LeakyRelu
2	BatchNormalization	-	-	-
3	Dense	512	-	LeakyRelu
4	BatchNormalization	-	-	-
5	Dense	8	-	tanh

TABLE 12. BiGAN discriminator's layers characteristics.

Layer Number	Type	Neurons	Input Dimension	Activation Function
1	Dense	1024	27	LeakyRelu
2	Dropout	-	-	-
3	Dense	1024	-	LeakyRelu
4	Dropout	-	-	-
5	Dense	1024	-	LeakyRelu
6	Dropout	-	-	-
7	Dense	1	-	Sigmoid

TABLE 13. Generator of Known/Unknown testing.

Layer Number	Type	Neurons	Input Dimension	Activation Function
1	Dense	512	27	LeakyRelu
2	BatchNormalization	-	-	-
3	Dense	512	-	LeakyRelu
4	BatchNormalization	-	-	-
5	Dense	8	-	sigmoid

IV. EVALUATION

Stratified 10-fold sampling strategy was used for evaluation. This strategy generates train and test sets at each split. When the training set is generated, it gets normalized and balanced as mentioned previously. At this point, different classifiers and GANs were trained on the training data.

Four metrics were used to evaluate the models: accuracy, recall, precision and F1-score. Given True Positives (TP), True Negatives (NP), False Positives (FP) and False Negatives (FN), equations of accuracy, recall, precision and F1-score are written in (4), (5), (6) and (7) respectively.

TABLE 14. KNN precision, recall and F1-score.

Class	Precision	Std	Recall	Std	F1-score	Std
Benign	0.89	0.0045	0.78	0.0043	0.83	0.0037
DDoS	1.00	0.0000	1.00	0.0000	1.00	0.0000
C&C-HeartBeat-Attack	0.48	0.0408	0.96	0.0246	0.64	0.0360
C&C-PartOfAHorizontalPortScan	0.02	0.0032	0.42	0.0623	0.04	0.0059
C&C-HeartBeat	0.21	0.0095	0.36	0.0118	0.26	0.0103
PartOfAHorizontalPortScan	0.94	0.0024	0.76	0.0050	0.84	0.0032
Okiru	0.43	0.0090	0.69	0.0132	0.53	0.0101
Attack	0.76	0.0118	0.92	0.0079	0.83	0.0079
C&C	0.39	0.0079	0.52	0.0099	0.44	0.0075

TABLE 15. RF precision, recall and F1-score.

Class	Precision	Std	Recall	Std	F1-score	Std
Benign	0.84	0.0037	0.73	0.0049	0.78	0.0038
DDoS	1.00	0.0000	1.00	0.0001	1.00	0.0000
C&C-HeartBeat-Attack	0.54	0.0516	0.83	0.0359	0.65	0.0405
C&C-PartOfAHorizontalPortScan	0.01	0.0026	0.30	0.0892	0.02	0.0050
C&C-HeartBeat	0.10	0.0058	0.18	0.0092	0.13	0.0070
PartOfAHorizontalPortScan	0.86	0.0038	0.67	0.0079	0.75	0.0055
Okiru	0.37	0.0070	0.50	0.0118	0.43	0.0075
Attack	0.83	0.0114	0.88	0.0089	0.86	0.0070
C&C	0.29	0.0112	0.47	0.0122	0.36	0.0116

TABLE 16. AAE + KNN precision, recall and F1-score.

Class	Precision	Std	Recall	Std	F1-score	Std
Benign	0.99	0.0100	0.99	0.0122	0.99	0.0100
DDoS	1.00	0.0000	1.00	0.0000	1.00	0.0000
C&C-HeartBeat-Attack	0.99	0.0164	1.00	0.0092	0.99	0.0100
C&C-PartOfAHorizontalPortScan	0.94	0.0739	0.93	0.0907	0.93	0.0751
C&C-HeartBeat	0.93	0.0843	0.93	0.0772	0.93	0.0789
PartOfAHorizontalPortScan	0.99	0.0092	0.99	0.0092	0.99	0.0092
PartOfAHorizontalPortScan	0.99	0.0092	0.99	0.0092	0.99	0.0092
Okiru	0.98	0.0264	0.98	0.0313	0.98	0.0290
Attack	1.00	0.0000	1.00	0.0030	1.00	0.0000
C&C	0.96	0.0439	0.96	0.0372	0.96	0.0408

TABLE 17. BiGAN + KNN precision, recall and F1-score.

Class	Precision	Std	Recall	Std	F1-score	Std
Benign	0.99	0.0122	0.99	0.0104	0.99	0.0100
DDoS	1.00	0.0000	1.00	0.0000	1.00	0.0000
C&C-HeartBeat-Attack	1.00	0.0090	0.99	0.0166	0.99	0.0100
C&C-PartOfAHorizontalPortScan	0.94	0.0764	0.92	0.0837	0.93	0.0748
C&C-HeartBeat	0.93	0.0789	0.93	0.0808	0.93	0.0815
PartOfAHorizontalPortScan	0.99	0.0092	0.99	0.0092	0.99	0.0092
Okiru	0.98	0.0284	0.98	0.0295	0.98	0.0290
Attack	1.00	0.0000	1.00	0.0030	1.00	0.0000
C&C	0.96	0.0395	0.95	0.0454	0.96	0.0408

The F1-score is used here because alternative metrics like accuracy are potentially misleading for imbalanced datasets which are common in anomaly detection tasks. Since F1-score is a geometric mean of recall and precision, this metric provides more meaningful results for imbalanced data sets.

$$Accuracy = \frac{(TP + TN)}{All} \quad (4)$$

$$Recall = \frac{TP}{(TP + FN)} \quad (5)$$

$$Precision = \frac{TP}{(TP + FP)} \quad (6)$$

$$F1 = \frac{2 \times Recall \times Precision}{(Recall + Precision)} \quad (7)$$

K Nearest Neighbor (KNN) and Random Forest (RF) were used for baseline comparison. Table 14 and Table 15 show the results of the precision, recall and F1-score of the KNN and the RF respectively.

Table 14 and Table 15 show that both RF and KNN did not perform well and resulted in F1-scores of as low as 0.02 in some instances.

The results for using Adversarial Autoencoder and BiGAN are shown in Table 16 and Table 17, respectively.

Table 19 shows that all metrics including accuracy, precision, recall and F1-Score were different across the

TABLE 18. Comparison of the various models based on average macro F1.

Model	Average Macro F1 Scores	Std
KNN	0.6019	0.0050
RF	0.5518	0.0055
AAE + KNN	0.9743	0.0182
BiGAN + KNN	0.9741	0.0186

TABLE 19. Statistical comparison of various metrics of the models (kruskal-wallis test; N = 10).

Metric	Test Statistics	p-value
Accuracy	32.986	0
F1-Score	32.934	0
Precision	32.943	0
Recall	32.933	0

TABLE 20. Performance metrics for the BiGAN to detect unknowns.

# features changed/ metric	1	2	4	8	16	27
Average Precision	1	1	1	1	1	1
Average Recall	0.81	0.95	1	1	1	1
Average Accuracy	0.89	0.97	1	1	1	1
Average F1-scores	0.85	0.94	1	1	1	1

various models. Pair-wise Mann-Whitney tests ($p < 0.05$) showed that AAE + KNN and BiGAN + KNN were not significantly different for all metrics while they were both different than RF and KNN for all metrics.

Table 18 summarizes the results. As can be seen using both AAE and the BiGAN had significantly better F1-Score than KNN or Random Forest.

Table 19 shows that all metrics including accuracy, precision, recall and F1-Score were different across the various models. Pair-wise Mann-Whitney tests ($p < 0.05$) showed that AAE + KNN and BiGAN + KNN were not significantly different for all metrics while they were both different than RF and KNN for all metrics.

Finally, the BiGAN and AAE models was also evaluated for the goodness of data being generated. The first evaluation was the Leave One Out (LOO) of KNN using Euclidean Distance as proposed by Guan *et al.* [57]. In this evaluation, real data labelled as 1 and GAN generated data labelled as 0 was passed to a KNN. For the AAE synthetic data was generated by using the decoder on random inputs from the latent space. The KNN (with distance = 1) was then trained on all data except for one instance that was used for testing. For both AAE and BiGAN, the accuracy of almost 100% meaning that the decoder and generator were not generating data from the same distribution. However, the mapping to the latent space thus generated was sufficient for a better classification than the original space.

The second evaluation was the GAN-train and GAN-train as proposed by Shmelkov *et al.* [58]. In this evaluation, the classifier, such as KNN, was used twice. The classifier was first trained on real train data and evaluated on GAN/AAE generated data. The KNN was also trained on generated data and tested on actual test data. In both AAE and BiGAN, the accuracy was almost zero supporting the conjecture that where the AAE and BiGAN were able to represent the data but could not generate data coming from the same distribution.

Table 20 shows that the BiGAN to detect unknown anomalies was very effective with F1-Scores of 1 as the number of mutations to features were increased. Even for a single mutation, the model had a decent F1-Score of 0.85.

V. DISCUSSION

As observed in the results section, the accuracies, and F1-scores of both AAE + KNN and BiGAN + KNN were almost same. Both models were able to recognize the DDoS and Attack. The classes C&C-PartOfAHorizontalPortScan and C&C-HeartBeat had the least F1-scores (0.93). This could have arisen from two reasons. The first reason is that both classes included devices connected to C&C server and the second reason is that both PartOfHorizontalPortScan and HeartBeat meant that the devices were being tracked but in different ways.

Because the .pcap files of the collected data are available, additional network analysers can be used to retrieve additional features. There is also more work to be done on increasing the number of instances of the minority classes. Another interesting aspect that could be done for anomaly detection is to find the shared features between different attacks and possibly detect new attacks that share those same features.

VI. CONCLUSION

The rapid increase in deployment of IoT devices has made them a dangerous unsuspecting participant in cyberattacks. This paper has shown that for a limited set of attacks and IoT devices, it is possible to use generative deep learning methods like AAE and BiGAN to classify attacks with a very high accuracy. Although there are several datasets regarding intrusion detection, it is better to use a dataset which was generated from IoT devices. Hence, in this paper we used a recent dataset called IoT-23. We implemented baseline models, Adversarial Autoencoders and Bidirectional GANs. Our results show that the GAN based models are more effective at identifying attacks and classifying them. We also tried randomizing the test set in a way that we can inject new information and the model was able to consider it as an anomaly.

ACKNOWLEDGMENT

This paper represents the opinions of the authors and does not mean to represent the position or opinions of the American University of Sharjah.

REFERENCES

[1] B. Alqahtani and B. AlNajrani, "A study of Internet of Things protocols and communication," in *Proc. 2nd Int. Conf. Comput. Inf. Sci. (ICCIS)*, Oct. 2020, pp. 1–6, doi: 10.1109/ICCIS49240.2020.9257652.

[2] W. Yang, ‘‘Research on network security problems and countermeasures based on the Internet of Things technology,’’ *J. Phys., Conf. Ser.*, vol. 1744, no. 4, Feb. 2021, Art. no. 042010, doi: 10.1088/1742-6596/1744/4/042010.

[3] S. Fenanir, F. Semchedine, S. Harous, and A. Baadache, ‘‘A semi-supervised deep auto-encoder based intrusion detection for IoT,’’ *Ingénierie Syst. Inf.*, vol. 25, no. 5, pp. 569–577, Nov. 2020, doi: 10.18280/isi.250503.

[4] C. Vorakulpipat, E. Rattanalerdnusorn, P. Thaenkaew, and H. D. Hai, ‘‘Recent challenges, trends, and concerns related to IoT security: An evolutionary study,’’ in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 405–410, doi: 10.23919/ICACT.2018.8323774.

[5] S. A. Butt, J. L. Diaz-Martinez, T. Jamal, A. Ali, E. De-La-Hoz-Franco, and M. Shoaib, ‘‘IoT smart health security threats,’’ in *Proc. 19th Int. Conf. Comput. Sci. Appl. (ICCSA)*, Jul. 2019, pp. 26–31, doi: 10.1109/ICCSA.2019.000-8.

[6] M. Burhan, R. A. Rehman, B. Khan, and B.-S. Kim, ‘‘IoT elements, layered architectures and security issues: A comprehensive survey,’’ *Sensors*, vol. 18, no. 9, Sep. 2018, Art. no. 9, doi: 10.3390/s18092796.

[7] A. Hameed and A. Alomary, ‘‘Security issues in IoT: A survey,’’ in *Proc. Int. Conf. Innov. Intell. Inform., Comput., Technol. (3ICT)*, Sep. 2019, pp. 1–5, doi: 10.1109/3ICT.2019.8910320.

[8] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, ‘‘Internet of Things: Security vulnerabilities and challenges,’’ in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2015, pp. 180–187, doi: 10.1109/ISCC.2015.7405513.

[9] K. Angrishi, ‘‘Turning Internet of Things (IoT) into internet of vulnerabilities (IoV): IoT botnets,’’ 2017, *arXiv:1702.03681*.

[10] R. Ahmad and I. Alsmadi, ‘‘Machine learning approaches to IoT security: A systematic literature review,’’ *Internet Things*, vol. 14, Jun. 2021, Art. no. 100365, doi: 10.1016/j.iot.2021.100365.

[11] S. Das, P. P. Amritha, and K. Praveen, ‘‘Detection and prevention of mirai attack,’’ in *Proc. Soft Comput. Signal Process.*, Singapore, 2021, pp. 79–88.

[12] B. Tushir, H. Sehgal, R. Nair, B. Dezfouli, and Y. Liu, ‘‘The impact of DoS attacks on resource-constrained IoT devices: A study on the Mirai attack,’’ 2021, *arXiv:2104.09041*.

[13] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, ‘‘DDoS in the IoT: Mirai and other Botnets,’’ *Computer*, vol. 50, no. 7, pp. 80–84, 2017, doi: 10.1109/MC.2017.201.

[14] *IoT connected devices worldwide 2019-2030*. Accessed: Jun. 17, 2021. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[15] *Cisco Annual Internet Report—Cisco Annual Internet Report (2018-2023) White Paper*. Accessed: May 31, 2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[16] S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour, ‘‘Intrusion detection systems in the Internet of Things: A comprehensive investigation,’’ *Comput. Netw.*, vol. 160, pp. 165–191, Sep. 2019, doi: 10.1016/j.comnet.2019.05.014.

[17] S. Garcia, A. Parmisano, and M. J. Erquiaga, ‘‘IoT-23: A labeled dataset with malicious and benign IoT network traffic (version 1.0.0),’’ *Zenodo*, vol. 20, p. 15, Jan. 2020, doi: 10.5281/zenodo.4743746.

[18] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, ‘‘Deep learning for anomaly detection: A review,’’ *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, Mar. 2021, doi: 10.1145/3439950.

[19] P. A. A. Resende and A. C. Drummond, ‘‘A survey of random forest based methods for intrusion detection systems,’’ *ACM Comput. Surv.*, vol. 51, no. 3, p. 48, 2018, doi: 10.1145/3178582.

[20] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, and L. Cui, ‘‘Robust detection for network intrusion of industrial IoT based on multi-CNN fusion,’’ *Measurement*, vol. 154, Mar. 2020, Art. no. 107450, doi: 10.1016/j.measurement.2019.107450.

[21] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, ‘‘A detailed analysis of the KDD CUP 99 data set,’’ in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.

[22] S. Latif, Z. Idrees, Z. Zou, and J. Ahmad, ‘‘DRaNN: A deep random neural network model for intrusion detection in industrial IoT,’’ in *Proc. Int. Conf. U.K.-China Emerg. Technol. (UCET)*, Aug. 2020, pp. 1–4, doi: 10.1109/UCET51115.2020.9205361.

[23] N. Moustafa and J. Slay, ‘‘UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),’’ in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[24] Y. Xu, Y. Tang, and Q. Yang, ‘‘Deep learning for IoT intrusion detection based on LSTMs-AE,’’ in *Proc. 2nd Int. Conf. Artif. Intell. Adv. Manuf.*, New York, NY, USA, Oct. 2020, pp. 64–68, doi: 10.1145/3421766.3421891.

[25] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, ‘‘Kitsune: An ensemble of autoencoders for online network intrusion detection,’’ in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.

[26] M. H. Shahriar, N. I. Haque, M. A. Rahman, and M. Alonso, Jr., ‘‘G-IDS: Generative adversarial networks assisted intrusion detection system,’’ 2020, *arXiv:2006.00676*.

[27] W. Fan, M. Miller, S. J. Stolfo, W. Lee, and P. K. Chan, ‘‘Using artificial anomalies to detect unknown and known network intrusions,’’ in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2001, pp. 123–130, doi: 10.1109/ICDM.2001.989509.

[28] S. P. RM, P. K. R. Maddikunta, M. Parimala, S. Koppu, T. R. Gadekallu, C. L. Chowdhary, and M. Alazab, ‘‘An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture,’’ *Comput. Commun.*, vol. 160, pp. 139–149, Jul. 2020, doi: 10.1016/j.comcom.2020.05.048.

[29] M. R. Shahid, G. Blanc, H. Jmila, Z. Zhang, and H. Debar, ‘‘Generative deep learning for Internet of Things network traffic generation,’’ in *2020 IEEE 25th Pacific Rim Int. Symp. Dependable Comput. (PRDC)*, Dec. 2020, pp. 70–79, doi: 10.1109/PRDC50213.2020.00018.

[30] M. Salem, S. Taheri, and J. S. Yuan, ‘‘Anomaly generation using generative adversarial networks in host based intrusion detection,’’ in *Proc. 9th IEEE Annu. Ubiquitous Comput. Electron. Mobile Commun. Conf.*, Nov. 2018, pp. 683–687, doi: 10.1109/UEMCON.2018.8796769.

[31] S. Huang and K. Lei, ‘‘IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks,’’ *Ad Hoc Netw.*, vol. 105, Aug. 2020, Art. no. 102177, doi: 10.1016/j.adhoc.2020.102177.

[32] I. Yilmaz, R. Masum, and A. Siraj, ‘‘Addressing imbalanced data problem with generative adversarial network for intrusion detection,’’ in *Proc. IEEE 21st Int. Conf. Inf. Reuse Integr. Data Sci. (IRI)*, Aug. 2020, pp. 25–30, doi: 10.1109/IRI49571.2020.00012.

[33] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, ‘‘Toward generating a new intrusion detection dataset and intrusion traffic characterization,’’ in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[34] R. Chauhan and S. Shah Heydari, ‘‘Polymorphic adversarial DDoS attack on IDS using GAN,’’ in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Oct. 2020, pp. 1–6, doi: 10.1109/ISNCC49221.2020.9297264.

[35] S. M. Lundberg, G. G. Erion, and S.-I. Lee, ‘‘Consistent individualized feature attribution for tree ensembles,’’ 2018, *arXiv:1802.03888*.

[36] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, ‘‘Adversarial autoencoders,’’ 2015, *arXiv:1511.05644*.

[37] S. Puuska, T. Kokkonen, J. Alatalo, and E. Heilimo, ‘‘Anomaly-based network intrusion detection using wavelets and adversarial autoencoders,’’ in *Proc. Innov. Secur. Solutions Inf. Technol. Commun.*, Cham, Switzerland, 2019, pp. 234–246, doi: 10.1007/978-3-030-12942-2_18.

[38] K. Hara and K. Shiomoto, ‘‘Intrusion detection system using semi-supervised learning with adversarial auto-encoder,’’ in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2020, pp. 1–8, doi: 10.1109/NOMS47738.2020.9110343.

[39] M. O. Kaplan and S. E. Alptekin, ‘‘An improved BiGAN based approach for anomaly detection,’’ *Proc. Comput. Sci.*, vol. 176, pp. 185–194, 2020, doi: 10.1016/j.procs.2020.08.020.

[40] J. Donahue, P. Krähenbáhl, and T. Darrell, ‘‘Adversarial feature learning,’’ 2016, *arXiv:1605.09782*.

[41] J. Donahue and K. Simonyan, ‘‘Large scale adversarial representation learning,’’ 2019, *arXiv:1907.02544*.

[42] S. K. Alabugin and A. N. Sokolov, ‘‘Applying of generative adversarial networks for anomaly detection in industrial control systems,’’ in *Proc. Global Smart Ind. Conf. (GloSIC)*, Nov. 2020, pp. 199–203, doi: 10.1109/GloSIC50886.2020.9267878.

[43] *Introduction to Scripting—Book of Zeek (Git/Master)*. Accessed: May 19, 2021. [Online]. Available: https://docs.zeek.org/en/master/scripting/intro.html#writing-scripts-connection-record

[44] V. Dutta, M. Chora, M. Pawlicki, and R. Kozik, "A deep learning ensemble for network anomaly and cyber-attack detection," *Sensors*, vol. 20, no. 16, Jan. 2020, Art. no. 16, doi: 10.3390/s20164583.

[45] *Base/Protocols/Conn/Main.Zeek—Book of Zeek (V4.0.1)*. Accessed: May 19, 2021. [Online]. Available: https://docs.zeek.org/en/lts/scripts/base/protocols/conn/main.zeek.html

[46] O. Faker and E. Dogdu, "Intrusion detection using big data and deep learning techniques," in *Proc. ACM Southeast Conf.*, New York, NY, USA, Apr. 2019, pp. 86–93, doi: 10.1145/3299815.3314439.

[47] C. Ieracitano, A. Adeel, M. Gogate, K. Dashtipour, F. C. Morabito, H. Larijani, A. Raza, and A. Hussain, "Statistical analysis driven optimized deep learning system for intrusion detection," 2018, *arXiv:1808.05633*.

[48] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50850–50859, 2018, doi: 10.1109/ACCESS.2018.2868993.

[49] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42210–42219, 2019, doi: 10.1109/ACCESS.2019.2904620.

[50] L. Zhang, M. Li, X. Wang, and Y. Huang, "An improved network intrusion detection based on deep neural network," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 563, Aug. 2019, Art. no. 052019, doi: 10.1088/1757-899X/563/5/052019.

[51] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, p. 27, Dec. 2019, doi: 10.1186/s40537-019-0192-5.

[52] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[53] F. Hussain, S. G. Abbas, U. U. Fayyaz, G. A. Shah, A. Toqeer, and A. Ali, "Towards a universal features set for IoT botnet attacks detection," 2020, *arXiv:2012.00463*.

[54] M. Hegde, G. Kepnang, M. Al Mazroei, J. S. Chavis, and L. Watkins, "Identification of botnet activity in iot network traffic using machine learning," in *Proc. Int. Conf. Intell. Data Sci. Technol. Appl. (IDSTA)*, Oct. 2020, pp. 21–27, doi: 10.1109/IDSTA50958.2020.9264143.

[55] A. Kumar, M. Shridhar, S. Swaminathan, and T. Joon Lim, "Machine learning-based early detection of IoT botnets using network-edge traffic," 2020, *arXiv:2010.11453*.

[56] F. Aloul, I. Zualkernan, N. Abdalgawad, L. Hussain, and D. Sakhnini, "Network intrusion detection on the IoT edge using adversarial autoencoders," in *Proc. Int. Conf. Inf. Technol. (ICIT)*, Jul. 2021, pp. 120–125, doi: 10.1109/ICIT52682.2021.9491694.

[57] S. Guan and M. Loew, "Evaluation of generative adversarial network performance based on direct analysis of generated images," in *Proc. IEEE Appl. Imag. Pattern Recognit. Workshop (AIPR)*, Oct. 2019, pp. 1–5, doi: 10.1109/AIPR47015.2019.9174595.

[58] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my GAN?" 2018, *arXiv:1807.09499*.

N. ABDALGAWAD (Student Member, IEEE) received the B.Sc. degree *(cum laude)* in computer engineering from the American University of Sharjah, United Arab Emirates, in 2020, where she is currently pursuing the M.Sc. degree in computer engineering. She is also working as a Graduate Teaching and a Research Assistant with the American University of Sharjah. Her research interests include machine learning, cyber security, field programmable gate arrays, and cloud computing. She is a member of Upsilon Pi Epsilon Honor Society and IEEE-HKN. She was a recipient of Abdulla Al Ghurair Foundation for STEM Education Scholarship. She won the Best Graduate Teaching Assistant Award for the year 2020 under the Department of Computer Science and Engineering, American University of Sharjah.

A. SAJUN received the B.S. degree in computer engineering, in 2020. He is currently pursuing the M.S. degree in computer engineering with the American University of Sharjah. He is also working as a Research Assistant in the field of deep learning, the Internet of Things, and smart solar energy systems. His research interests include deep learning, the Internet of Things, automated wildlife monitoring, and smart energy.

Y. KADDOURA (Student Member, IEEE) received the B.Sc. degree (Hons.) in computer engineering and a minor in computer science from the American University of Sharjah (AUS), in 2019, where she is currently pursuing the M.Sc. degree in computer engineering. She is also working as a Research Assistant in fault diagnosis in finite state machines. Her research interests include cloud computing, machine learning, and cybersecurity. She is a member of the Upsilon Pi Epsilon Honor Society and the IEEE-HKN Honor Society.

I. A. ZUALKERNAN (Member, IEEE) received the B.S. (Hons.) and Ph.D. degrees in computer science from the University of Minnesota, Minneapolis, in 1983 and 1991, respectively. He was an Assistant Professor with the Computer and Electrical Engineering Department, Pennsylvania State University, from 1992 to 1995. He was a Principal Design Engineer with AMCS Inc., Chanhassen, Minnesota, from 1995 to 1998. He was the Chief Executive Officer of Askari Information Systems, from 1998 to 2000, and the Chief Technology Officer of Knowledge Platform, Inc., Singapore, from 2000 to 2003. In 2003, he joined the American University of Sharjah, United Arab Emirates, where he is currently a Professor in computer science and engineering. He is the author or coauthor of more than 200 peer-reviewed articles and has received the 2020 IEEE Consumer Electronics Society Chester Sall Award. His research interests include consumer systems, sensor-based internet applications, AIoT for consumer devices, and the Internet of Things (IoT).

F. ALOUL (Senior Member, IEEE) received the B.S. degree *(summa cum laude)* in electrical engineering from Lawrence Technological University, Michigan, USA, and the M.S. and Ph.D. degrees in computer science and engineering from the University of Michigan, Ann Arbor, USA. He is currently a Professor and the Department Head of Computer Science/Engineering and the Director of the HP Institute, American University of Sharjah (AUS), United Arab Emirates. He has more than 130 publications in international journals and conferences, in addition to one U.S. patent. His current research interests include cyber security, mobile applications, and design optimization.

He received a number of awards, including the Global Engineering Deans Council (GEDC) Airbus Engineering Diversity Award, the Sheikh Khalifa Award for Higher Education, the AUS Excellence in Teaching Award, the Abdul Hameed Shoman Award for Young Arab Researchers, and the Sheikh Rashid's Award for Outstanding Scientific Achievement. He is a regular invited speaker and panelist across a number of international conferences related to cyber security, technology, innovation, and education. He is also a Certified Information Systems Security Professional (CISSP).